JPL D-1413

Autonomous Spacecraft System Control Topics

P.R. Turner

15 March 1984

Prepared for

Office of Aeronautics and Space Technology
RTOP 506-64-15

National Aeronautics and
Space Administration

JPL

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

FOREWORD

Design of spacecraft for deep-space planetary exploration has led the Jet Propulsion Laboratory to develop spacecraft controlled by complex automated command sequences which include the capability for on-board redundancy management and fault protection. This design feature allows the spacecraft to operate independantly from real-time ground control, and with reduced risk to mission success. The combination of automated operation and fault protection provides the attribute of operational spacecraft autonomy. This attribute has been recognized as a desirable goal for a variety of military and civilian space applications and several years of effort have been expended in the definition of autonomy and the analysis of appropriate design methodologies. The work has produced a great deal of insight into the nature of spacecraft autonomy and this document summarizes many of the important points in a topical fashion.

The body of the document is organized into a series of 7 major topics with several paragraphs of discussion for each. The appendix provides a set of concise statements of the major points in each of the topical areas.

TABLE OF CONTENTS

PRECEDING PAGE BLANK NOT FILMED

INTENTIONALLY BLANK

TABLE OF CONTENTS (CONTINUED)

INTRODUCTION

Jet Propulsion Laboratory involvement in Earth orbiting spacecraft autonomy began in the summer of 1980 with a workshop sponsored by the Air Force Office of Scientific Research involving industry and the Academic Community. The objective was to define the concepts and technology needed to increase the automation of spacecraft operation and reduce the dependance on ground control. The motivation for this activity was simple. JPL spacecraft designed for planetary exploration missions incorporated a degree of automated, fault-tolerant operation that exceeded the current capability of earth orbiting spacecraft for civilian and military applications. Future military needs include the requirement for spacecraft to survive the occurrence of faults and continue normal operations for extended periods without ground control. The potential for reduction of ground support requirements for significant periods of time provides the potential for significant life-cycle cost savings beyond the direct mission operations requirements.

This initial effort led to the establishment of the Autonomous Spacecraft Program at JPL. Development of design concepts for autonomous spacecraft control has continued with the consideration of autonomy requirements of NASA missions as a significant product. Planetary class missions have been considered, and the Space Station Program provides an additional dimension in complexity and mission diversity. The autonomy considerations developed in other mission contexts have been extended to the set of program elements and mission areas that comprise the space station program concept. This document summarizes a set of insights into the nature of autonomous spacecraft control that spans the scope of missions from single spacecraft missions to complex systems with man as an essential on-board element.

SECTION 1

AUTONOMY DEFINITION AND ATTRIBUTES

Detailed discussion of autonomy is not possible without a specific definition of the term. This concern and the differentiation between autonomy and automation has been a major point of issue from the beginning of the work in this area. The following definition has been refined through several interations.

1.1 DEFINITION OF AUTONOMY

The following definition of autonomy describes a fundamental system level attribute of a spacecraft system:

> "Autonomy is that attribute of a system that allows it to operate without external control and to perform its specified mission at an established performance level for a specified period of time."

There are several fundamental points contained within this definition:

(1) The "system" whose autonomy is being described must be clearly bounded to allow differentiation between the system and its external interfaces.

(2) The definition implies a control process whose functions are carried out within the bounds of the autonomous system. Human resources may be utilized in the control process if the system boundary includes a manned component. The term "machine autonomy" has been used to describe those circumstances where human resources are not normally included in the control process.

(3) The "specified mission" must be defined for each individual project, program, etc. that has autonomous operations requirements. The portion of the project-specific mission requiring autonomous operation may be the entire mission or only a specific portion of the nominal mission.

(4) The "established performance level" may include a specification of full nominal performance or some minimum level of performance that is adequate to satisfy mission requirements for the autonomous operation period.

(5) The "specified period of time" is necessary to scope the autonomous control problem and to ensure that the proposed implementation meets this requirement.

(6) Autonomy implies adaptability in the control process. This adaptability includes the ability to continue to operate at some level of performance in the presence of faults (fault-tolerance, redundancy management) and to maintain the specified level of system performance (calibration, health maintenance). Mission-specific requirements for adaptability of the onboard control process may include the ability to select alternate control modes and sequences, to perform navigation functions, and/or to collect and process mission data.

1.2 AUTONOMOUS CONTROL CHARACTERISTICS

Autonomy requires a three-step control structure. Those processes that require closed-loop control may cycle from step (3) back to step (1), below.

(1) Sense and analyze the state of internal or external quantities which are inputs to the control process.

(2) Derive and command a response by the system that meets an appropriate objective.

(3) Act to implement the response.

The control process is implemented through control resources that imply a connection between command resources and data management resources. Sensory data required by the command resource may be collected and communicated by data management resources in a manner normally used for engineering telemetry. A conventional command system may be utilized by an autonomous command resource to implement desired system and subsystem level state and operating mode changes. Programmable computer resources for the logical direction of the control process allow the flexibility needed to provide adaptable control. As system complexity increases, distribution of separate control resources to the subsystem level and below: 1) serves to reduce the pressure of multiple demands upon a central resource, 2) reduces the interdependence of subsystems, and 3) supports the evolvability of the system to meet new requirements.

1.3 HISTORICAL DEVELOPMENT

Early planetary exploration spacecraft were designed as semi-automated systems. Hardwired sequencers controlled payload functions on the basis of a timer initiated in the launch phase of the mission. Trajectory correction maneuvers which maintained the nominal timeline of the sequencer were ground initiated and automatically executed by on-board hardwired controls. Increasing mission complexity with attendant payload and spacecraft control requirements led to the provision of in-flight programmable sequencing and control that was eventually supplied by on-board digital computers. Additional mission complexity results in increased risk of failure which may be countered by dedication of some portion of the on-board control resource to fault-protection. The existence of a programmable control resource with access to the engineering data of the spacecraft allows closed loop control for both fault protection/redundancy management and maintenance of the operating condition of subsystem components.

The preceeding description is an example of an automated system evolving into an autonomous system. That is, an automated system functions within a limited and pre-defined scope of circumstances. The addition of fault-tolerance and the ability to adapt control behavior to changing external and internal conditions on the basis of sensed information leads to an autonomous system.

1.4 AUTONOMY/AUTOMATION EXAMPLE

A further differentiation between the two terms is illustrated by the following examples. Attitude control of three-axis stabilized spacecraft is typically carried out by a closed-loop control process. Current designs provide a series of automated operating modes of the control system to accomplish specific mission phases. Sun acquisition, earth acquisition, and nominal on-orbit nadir pointing are typical examples of automatic control operating modes. These normal operating modes may be perturbed by the occurrence of component or sensor failure. Such an external disturbance usually results in a fail-safe condition that requires external control action to restore normal operation. An autonomous attitude control function would, at a minimum, attempt to restore the normal operations through a combination of redundancy management and reacquisition of attitude references. Failure in this attempt at autonomous recovery would result in no worse than a safe spacecraft state. The autonomy in the system would be designed to restore normal operating conditions within the bounds of availiable resources.

It is important to note that automated functions are used in the achievement of autonomy. Autonomous control extends the automated functions by providing the system with enhanced adaptability and fault protection.

1.5 THE SCOPE OF AUTONOMY

The scope of autonomy in a system is a significant issue which becomes critical when the system is as large and complex as the space station. The scope of autonomy has at least two dimensions, those of function and time. Functional autonomy identifies those functions which will be controlled within the system bounds and the required level of performance in autonomous operation. The time scope includes both the duration of autonomous operation and the distribution of control tasks in time.

1.6 FUNCTIONAL SCOPE OF AUTONOMY

Functional candidates for spacecraft autonomy have been described in four areas (Reference 1-1). These areas are system health and maintenance, navigation, payload sequence generation and control, and payload data processing. These generic areas rapidly expand in complexity as lower level mission functions are enumerated and described (Reference 1-2). The issue is further complicated by the fact that many of the more detailed functions in these categories are interdependent. Health and maintenance of the system is fundamental to all system operation. Though the function of an individual component may not be critical to overall system performance, it can still affect the operation of some part of the payload and its command sequencing. Similiarly, payload design and the sequencing of payload operations will place varing

demands upon the operation of the core station functions of power, attitude control, data management, etc. that provide utility support for the payloads. These functionally interdependent areas are significant design issues for autonomous spacecraft systems.

1.7 TIME SCOPE OF AUTONOMY

The time scope of autonomous operations has two aspects. The first is the duration of autonomous operation for a function. Does the function have to operate without outside control for a period of an hour, a day, a week, the station lifetime? This may be both a characteristic of the function itself and a reflection of the mission requirements for the function. Real-time control of an Orbital Maneuvering Vechicle (OMV) docking maneuver may require autonomous control by the orbiting station, but other aspects of configuring the OMV or station proximity operations subsystem may offer potential trade-offs for ground support. This leads to the second time aspect of autonomy-segmentation of tasks with respect to real-time requirements. A given function will consist of a series of tasks ranging from planning and command sequence generation, through real-time task control, to analysis of results. The nature of these tasks and the difficulty of performing them autonomously may vary greatly. Planning, scheduling, and sequencing activities tend to be well suited to application of Expert Systems or Artificial Intelligence techniques while low-level control actions tend to require deterministic responses to sensed data.

1.8 TYPES OF AUTONOMOUS FUNCTIONS IN A SYSTEM

Autonomous control function candidates are located in many system areas and involve different enabling technology areas. The space station provides a wide range of potential applications to meet desired operational goals (Reference 1-3). Core station operation provides one set of possibilities involving autonomous spacecraft design methodology with direct connections to computer hardware and software technologies. Major mission functions of the space station such as Orbit Transfer Vehicle (OTV) servicing and operation, OMV servicing and operation, and proximity operation with space shuttle involve specialized technologies of robotics, teleoperation, and Expert Systems/Artificial Intelligence. These specialized technologies may also have applications to the core station autonomy implementation. There is a natural tendancy to lump all these complex functions and enabling technologies under a central banner of autonomy/automation. This would seem to simplify issues by taking advantage of the similiar high level requirement of reducing manned involvement in system control. This commonality does not extend too far down into the system design, however. The real commonality lies in the need for computing and data processing (processing, storage, and communications) hardware and the software and system design methodologies that are needed for control implementation. These are the basic tools to construct the "analyze" and "initiate response" portions of the three step control process described in subsection 1.2. The "sense" and "act to implement response" portions are significant and often unique parts of the individual function being controlled.

1.9 AUTONOMY TECHNOLOGY GOALS AND LEVERAGE

The autonomous control of core space station functions (executive control, power, attitude control, etc.) are fundamental to the space station operation over its lifetime. The technologies needed to implement this control are design methodologies, computer hardware technologies, and software technologies that are equally applicable to specific payload or service functions such as OMV operations or propellant loading. The largest leverage for autonomy/automation technology is to develop and demonstrate technology and design options for core station functions that will be in continuous operation throughout the space station life and support increased crew productivity in day-to-day station operation. The technologies that support this can be combined with specialized sensor and control requirements of more restricted disciplines and lower-level functions to support autonomy/automation of these less commonly used service functions of the space station.

SECTION 2

AUTONOMY AND COST/BENEFITS ISSUES

Addition of autonomy to a space system design increases the cost and complexity of the space segment above that of a conventional design where provisions for automated fault detection and correction are absent. The increase in cost and real benefits derived from autonomous control are a unique function of each mission and its requirements. The following points deal with some of these issues.

2.1 AUTONOMOUS OPERATIONS VERSUS NON-AUTONOMOUS

Simple missions that may be satisfied with preprogrammed control strategies or which have limited, predetermined objectives may be implemented without complex control resources that characterize current design concepts for autonomy. Addition of complex programmable control resources with the attendant software development and system test requirements significantly increase costs. Simple design techniques for implementation of fault protection could provide limited autonomy of operation and the attendant benefits without requiring more complex design for adaptable control on-board. Satellites which were developed in the Initial Defense Communications Satellite Program during the 1960's are examples. These satellites had a nondirectional communications repeater, no ground command capability, and redundancy provided through constellations of simple spacecraft. They were totaly automated but only autonomous in the aggregate of many spacecraft since a failure of a primary component caused loss of mission capability on an individual spacecraft.

More complex missions have evolved to require programmable control resources for the spacecraft as a system and to allow the payloads to be operated by pre-determined command sequences. These more complex spacecraft already pay for the inclusion of significant on-board redundancy, sensing of the operating state and health of components, and the organization of the sensor data for transmission to ground control facilities. A design that utilizes these basic resources to analyze the sensor data on-board, direct the selection and execution of stored contingency command sequences, and reconfigure the spacecraft to meet the changing conditions can provide significant autonomy for its operations. Use of resources in this fashion and cost of additional resources for evolution of new capability may be less than that required to add similiar adaptability to the simpler missions of the type mentioned above. The cost of basic control features required on-board for implementation of a normal mission should not be "book-kept" against autonomy, even though the resources can be designed to implement autonomy.

2.2 DATA POINT - AUTONOMY ADDED TO EXISTING DESIGN

A major aerospace contractor is reported to have conducted a study of adding autonomous control features to an existing design which was not considered autonomous in health and maintainance. The details of the effort and program were proprietary, but the projected increase in spacecraft cost over the existing design was approximately 10 percent to add a significant amount of autonomous control.

2.3 DISTRIBUTION OF COSTS AND BENEFITS IN A PROGRAM

Cost increases due to implementation of autonomy tend to appear at the beginning of a program. Design, construction, and integration test of complex spacecraft containing embedded computers for on-board control will increase as the functional requirements increase. On-going software development and maintenance costs in operations may increase if there is no significant need for these in a non-autonomous design implementation. The major benefits of autonomy tend to occur in operations and later in program life. Reliable and properly designed autonomous control can reduce the need for ground support operations and significantly reduce the risk of loss of a spacecraft due to failures (Reference 2-1). These benefits can not be achieved unless they are sold to program management and operations plans are developed to take advantage of them.

Cost differences and potential benefits may thus be characterized as "front-end loaded with life cycle payoff". The main difficulty with this arrangement is that a program with front end cost difficulties may choose to solve their front end problems by reducing costs associated with autonomy while accepting the increased life-time operating costs.

2.4 AUTOMATION OF GROUND OPERATIONS

Addition of on-board computer resources and the attendant flight software development requirements has impact upon the ground operations facilities and personnel. This impact may be minimized in the lifecycle by appropriate ground system design. Automated telemetry and spacecraft state analysis can improve productivity of ground personnel in normal health and maintenance support. Well designed software development and test facilities can minimize software costs whether associated with autonomy or normal operational requirements. Visibility requirements that specify on-board recording of autonomous control actions (audit trail) support the automated processing of spacecraft operation data on the ground and simplify the ground support task for autonomous spacecraft. Ground operations cost savings must be achieved by utilizing the fault-tolerant nature of autonomous spacecraft design to reduce operations loads while automating remaining operations to achieve maximum productivity of ground personnel.

2.5 SYSTEM RESOURCE MARGINS FOR AUTONOMOUS CONTROL

Fault-tolerance for autonomy may be increased by a "tall-pole in the tent" approach. This is characterized by identifying specific high-risk areas in a design and providing specific modifications or software algorithms to meet the individual problem. To some degree, this will be the reaction to newly perceived problems in any design activity. Recognition of fault-tolerance as an autonomous system design requirement and the provision of control resources with significant capability margins (memory, throughput, etc.) at specific points in the design process can serve to offset the occurrence of late recognition of requirements. However, selection of the margins should consider the potential for increased resource requirements for normal autonomous control features as well as fault-tolerance related "fixes". Evolution of understanding of a design during the design process tends to significantly increase the complexity of control design for nominal operations

as well as for faults. As a net result, a variation on Parkinson's law expressed as "requirements will expand to fill the resource availiable" tends to be a bit optimistic if anything. Project Galileo established memory and performance margin goals that appeared conservative at the beginning of the program. These were quickly overrun at an early point in the design, in spite of the management attention which was focused on the issue.

2.6 AUTONOMY AND RELIABILITY

Reliability modeling techniques currently in use are not applicable to calculating the reliability of systems incorporating autonomous control. Probabilistic models that include reliability improvements due to redundancy do not allow for reliability of the control process. The ability to correctly assess a fault and command the switching to the redundant elements in time to prevent significant impact on the system mission is assumed to occur with 1.0 reliability. These model features are not unreasonable for assessment of reliability of systems where control architecture and distribution of control between on-board and external resources is not an issue. The models are clearly inadequate for assessing autonomous systems reliability. Autonomy adds on-board complexity to a system and the added hardware and software components to implement control will always have reliabilities less than 1.0. However, the autonomous system should be assessed against the total ground and space segments of a non-autonomous system to compare system "operability" and "availiability", rather than only conventional reliability.

Autonomous control includes fault tolerance in the system and in the control resource. Its primary contribution to the system reliability is to detect and correct faults before they can have significant impact on the system, and maintain it in operating condition with fail-safe provisions for those conditions which cannot be corrected with existing resources. Modeling of the contribution to system reliability must include the reliablity of the fault detection and correction process as a minimum. The time aspect of real-time monitoring and control on-board versus the delays implicit in external ground control schemes should also be factored into a model. Finally, reliability factors to account for ground control proceedural errors should be included.

SECTION 3

SYSTEM ARCHITECTURE FOR AUTONOMOUS CONTROL

The three step control structure logic described in subsection 1.2 implicitly links autonomous control with requirements for state sensing, data processing, data communications, and command/control of functional system elements. Design of a system for implementation of autonomous control must consider the functional distribution of control logic and the required data communications amoung functional elements in the development of a data system architecture. Reference 2-1 describes a candidate functional architecture for the autonomous control of core space station functions. This section summarizes some of the points discussed in the reference.

3.1 HIERARCHICAL STRUCTURE OF CONTROL FUNCTIONS

The traditional division of a defined system into subsystems and elements within the subsystems leads to a hierarchical control structure. Subsystems and their elements provide services through internal functions which may be interactive with external functions or which may be controlled locally without external impact. The local or interactive nature of control for a function may well be a design option in its implementation. The design of distributed system control is simplified by decoupling the lower-level functions from one another as much as possible. The decoupling of functions also simplifies integration testing of the system and evolutionary addition of new capability or upgrading of baseline elements.

Figure 3-1 is taken from Reference 2-1 and depicts a candidate functional control hierarchy for space station core functions. The levels of the hierarchy are numbered with level 0 representing the executive control functions required for system control and crew/ground interface functions. Traditional functional subsystems have executive control functions at level 1 which allow for direct external control and which interface with the level 0 station executive functions. Level 2 represents the control and execution of internal applications performed by the subsystems. Lower levels are provided for control sensors and "smart" sensors, actuators, etc. which implement the functions of the subsystems.

3.2 DATA COMMUNICATIONS AND CONTROL DISTRIBUTION

Data flow consists of sensory data for control of applications functions (attitude control, power regulation, etc.), sensory data for health, maintenance, and system status monitoring, and command traffic (controller intercommunciations, controller to device, etc.).

Distribution of control resources and command capability to lower levels of the hierarchy can reduce the data rate requirements for automated control. A lower level controller can operate with whatever control bandwidth is appropriate for its tasks and communicate status information to higher level controllers at a reduced rate. Local storage of complex command sequences can allow higher level controllers to direct major low level activity with simple commands that select the desired low level sequences. An example of this

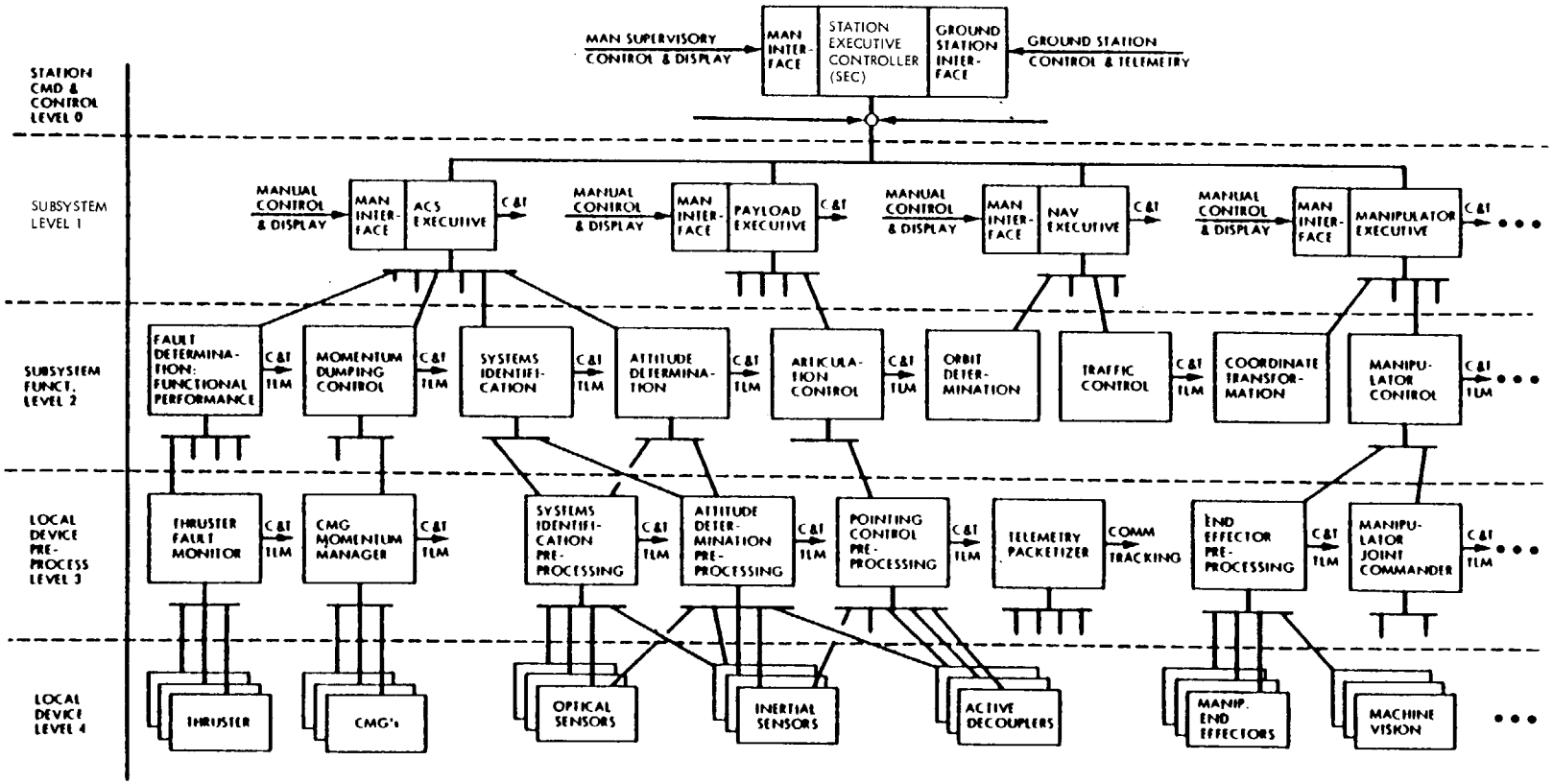

Figure 3-1. Autonomous Command and Control Architecture

technique in current practice is the design of complex sensors and "smart devices" with embedded processors. These elements of the lower level of the functional hierarchy commonly have complex local control sequences implemented in Read-only Memory (ROM). Control resources at higher levels of the hierarchy can command specific "operating modes" of the sensor or device without direct access to the detailed local commands that establish the modes.

Such design techniques can reduce the inter/intracommunications data rate requirements for autonomous control. Reduced data rates simplify the fault-tolerant data communications bus for control functions. The multi-hundred megabit data-rate requirements proposed for some space station payloads and for specialized services such as video can be provided by a separate high speed bus whose fault-tolerance is not as critical to station operation and whose resource will not be impacted by contention with high priority control data.

3.3 RESOURCE CONTENTION

The distribution of control resources and responsibilities to low levels of design simplifies the design and implementation of control in a complex system. Processes can execute concurrently at different control bandwidths and not compete for processing time in a common resource. The process of designing interdependent functions can concentrate upon processing and data interchange characteristics with contention for data communications as a primary concern.

3.4 FAULT TOLERANCE IN THE ARCHITECTURE

Designing fault tolerance into the system can allow continued operations in the presence of faults. This can minimize anomalous system behavior which may require immediate analysis and intervention to prevent loss of system performance or more serious consequences. In addition, the affects of a single fault may be prevented from propagating to other parts of the system where cascading impacts upon system operation mask the origional fault. Faults may occur in the control resource components or in the basic subsystem elements.

Faults in the control resource components can produce a variety of serious consequences. At best, there will be a loss of control capability for some period of time until the problem is identified. At worst, the control elements may propagate improper actions to other parts of the system and the problem may appear as anomalous behavior of other components. Minimization of fault impact in this vital area of control can be supported by fault-tolerant, self-checking control resources. At a minimum, fault-tolerance is required in the level 0 station executive controller to protect the top level of interactive control and the interface to flight/ground crew. The next candidate is resources for critical level 1 subsystem executive functions. Lower-level fault-tolerance may be supported by the higher-level executives or distributed to levels whose functions are critical in terms of safety, system impact, or timing.

Functional applications below the subsystem level are generally provided through redundant elements and sensing of operational status parameters to support failure detection. Executive control resources can use the sensed status data to detect faults and initiate replacement with redundant elements. Many faults can be prevented from affecting external functions in this manner, and those that do impact other functions may be isolated. The local control resource provides notification of the fault location to other control resources to prevent them from mistakenly perceiving potential faults in the responses of their respective status input parameters. Some faults in functional areas with significant external interactions may require supporting reconfiguration or reinitialization of external subsystems. This interactive response can be coordinated by the executive resources at higher functional levels.

SECTION 4

AUTONOMY AND ARTIFICIAL/MACHINE INTELLIGENCE

Machines that perform as a human would perform in the same circumstances have long been a challenging technology goal. The technology for achieving this behavior is usually classified as Artificial Intelligence. The actual implementation of this technology as part of a system results in the attribute of machine intelligence. Machine intelligence may be implemented as part of system automation. It can contribute to system autonomy and reduce crew workload in planning and supervision of automated functions. This section focuses on implementation of machine intelligence through Expert System (ES) technology.

4.1 EXPERT SYSTEM DEFINITION

An expert system addresses a specific field of knowledge. Within that field (the "Domain" of the ES) the decision making and control performance of the expert system attempts to approximate the best performance of a human expert in the field and (hopefully) may surpass the average performance of a given set of experts.

Applications of ES technology are commonly implemented in what is called a knowledge-based system. Expert knowledge in the system is organized into three areas: data, knowledge base, and control (Reference 4-1). The data represents declarative knowledge about the task being solved and the state of the system in solving the problem. The knowledge base provides the generic description of the problems that lie in the domain of the system's expertise. It consists of those informational relationships and procedures that might possibly be applied to the data to produce a problem solution. The control structure is the programmed logic that applies the knowledge base to the particular problem described by the data in order to yield a solution.

4.2 EXPERT SYSTEMS DOMAINS AND SPACECRAFT CONTROL

The domain of an ES is the specialized area of knowledge represented in its data, knowledge base, and control structure. The best known applications in literature are for medical diagnosis, analysis of geological survey data, and computer system configuration generation. Recent work and work in progress includes applications for tactical aircraft combat mission planning (Reference 4-2), spacecraft power subsystem control, teleoperations control (Reference 4-3), and command sequence generation for spacecraft control (Reference 4-4).

These latter areas show promise for autonomous spacecraft control applications. The earlier work in the field has concentrated on the use of ESs as diagnostic aids to humans whose expertise might not be as broad as that which has been incorporated in the machine. These systems may not provide satisfactory solutions to all problems in their domain, or may not provide a solution at all. These off-line advisory characteristics are not acceptable as part of a real-time control system, which is one of the more challenging

requirements for autonomous spacecraft design. Real-time control, planning/scheduling of resources, and fault identification and correction at system level are some major functions that can benefit from application of ESs methodology. Some areas for development of ESs relevant to control of complex spacecraft systems include:

(1) Definition of spacecraft system control domains.

(2) Development of subsystems as components of a distributed, hierarchical control system.

(3) Development of subsystems as components of a real-time, closed loop control system for spacecraft functions.

(4) Development of interface methodologies for multiple expert systems which support interdependent functions.

(5) Use of audit trail analysis or other techniques in expert systems to provide increased visibility into system operation for test and validation.

(6) Use of Artificial Intelligence (AI) and ES techniques in multi-mission or mission adaptable software design for flight applications.

(7) Development of techniques for reducing overhead for general knowledge base implementation to conserve on-board memory resources.

(8) Development of search techniques that reduce trial search times in specific spacecraft oriented applications and improve performance in support of real-time applications.

(9) Development of a natural language interface for spacecraft command, control, and sequencing.

(10) Incorporation of (9) as part of a distributed "operating system" for spacecraft control.

4.3 COMPLEXITY AND RESOURCE USAGE

General-purpose ESs require large computer resources. Resource-inefficient coding languages, knowledge bases containing redundant information, and heuristic techniques for problem solving place significant burdens on spacecraft systems constrained by size, mass, power, and timeline actions. Specialized processors for AI/ES development indicate the nature of this problem. A configuration of the Symbolics 3600 LISP machine is advertised as having 474 megabytes of physical storage and a 1 Gigabyte virtual memory capacity.

Throughput is a significant resource in control systems. Applications which require extensive searching or problem solving may not provide sufficient throughput for real-time control applications. Those techniques

with non-deterministic or variable execution times may not be acceptable in applications that involve high-bandwidth processing or which interface with other processes where execution timing is critical for synchronization of the processes.

Advances in computer hardware technology will reduce size, weight, and power requirements of resources, but system designers will need to simplify control requirements. Ground based systems for research, analysis, and software development can be used to reduce the knowledge base resource requirements and domain of on-board expert systems. This may be accomplished by performing a portion of the function on the ground (i.e. long term planning) or by compiling a software system for use in a more limited on-board resource.

A hierarchical and distributed control system can reduce domain and resource requirements on an ES by locating direct control of detailed tasks within lower-level resources. The ES then is responsible for scheduling and planning the sequencing of the lower-level tasks. Further discussion of the topics of control distribution and time partitioning of functions is presented in a later topic of this section.

Reference 4-5 proposes an ingenious approach to ES implementation that could be significant for flight applications. The more volatile portions of the ES - the knowledge base and the data base would be implemented as software components in random-access memory. The control portion, including the inference engine, would be implemented as a special purpose processor chip which would be optimized for symbolic processing. This approach could include the design of the symbolic processor as a "co-processor" in a family of flight -qualified, special-purpose processors with suporting input/output functions. The Intel 80XX family of processors provides an example with the 8087 Numeric Co-Processor providing high-speed floating point mathematics support to the basic 8086/8088 processors. This approach to providing specialized symbolic processing might be a powerful means of "shrinking" current large, general-purpose symbolic processing computers such as the Symbolics 3600 mentioned above.

4.4 EVOLUTION OF AUTONOMOUS CONTROL SOFTWARE TOWARDS EXPERT SYSTEM STRUCTURE

Software for autonomous control functions was first used in planetary missions to provide fault protection for specific spacecraft design features. This application required the design and coding of a separate algorithm for each problem. An algorithm may vary from very simple to very complex. It may address only a limited number of faults or may provide functional protecton against a wide variety of faults. Routines become very numerous as the spacecraft design grows more complex. The increasing amount of fault protection software within a spacecraft design has led to the concept of a "fault protection subsystem" composed of this software. This "fault protection subsystem" concept is an organizational approach to placing responsibility for fault protection design and the implementation of the software portion in a specific system engineering group. The actual provision of fault protection as an attribute of the system requires trades between

system and subsystem level design features and is not easily divisible into a seperate "subsystem". This evolving process has been extended to the design of algorithms and associated software routines to implement individual autonomous control functions without regard to their support for fault protection. This methodology has been referred to as "algorithmic autonomy" to differentiate it from AI related approaches to control. This should not be confused by the detail that expert systems and other AI applications also rely upon algorithms within their software structure.

The growing complexity of software for autonomous control produces a significant problem in software modification and testing. It becomes difficult to design and test an initial baseline configuration and the problem does not improve later as the baseline is modified and tested in flight. "Data base driven" designs have been developed to partially alleviate this problem. The control logic of the software is designed around variables in a data base that allow the logic to be changed by updating the data base rather than recoding. This removes a large number of problems that can arise when software is recompiled and relinked into a new configuration. The ability to change the logical behavior of the software by changing the data base values simplifies testing by limiting the portions of the software that interact with the changed logic. Regression testing of the remainder of the logically unaffected software is not necessary as it would be the case for a new recompiled and relinked software package.

This technique of using data base, value driven logic can be extended to include a relational data base of generic logical routines, which are selected by a executive or control routine. The appropriate logical routine is selected by input state and data base values. This software engineering technique begins to approach the implementation of a knowledge-based expert system. The data base contains the state and control parameters equivalent to the data base of the ES. The generic logical routines form a collection of capabilities similiar to a knowledge base, and the executive control software provides a control structure to drive the selection and operation of routines in the "knowledge base" based upon input and data base content. Current approachs to software design for autonomous control can thus evolve towards an ES structure. This approach may make it difficult to incorporate the full power of generic techniques developed in ESs work. On the other hand, it may produce a system with the desired domain of expertise without some of the penalties of ES overhead mentioned in subsection 4.3 above.

4.5 EXPERT SYSTEMS IN A HIERARCHICAL CONTROL STRUCTURE

There is a tendency to use the term "ES" as a convenient buzz word to signify a new approach to implementation of complex systems. ES techniques have great potential for enabling solutions to complex planning and control problems in automated and/or autonomous systems. It is necessary, however, to consider that ES applications are only a part of the overall functional system. A hierarchical structure is a fundamental characteristic of autonomous control implementation (Section 3.) and ESs techniques are poorly suited for the lower, more detailed levels of the control structure. It is necessary to define the appropriate scope of ES implementation in a control structure and clearly

define the bounds of responsibility for the "ES". Failure to accomplish this will lead to confusion over the actual value of ESs in control applications. Also, it can lead to needless expenditure of resources by attempting to force application of ESs techniques on problems which are more efficiently solved by traditional software techniques.

The appropriate issue is to consider the characteristics of the hierarchical control structure for an application and identify the appropriate scope of responsibility for ESs techniques. Principal characteristics of interest are the increasing level of detail in the lower levels of the hierarchical control structure, the functional characteristics of the various levels, and the time domain of interest in the control problem.

4.6 CONTROL CHARACTERISTICS IN THE FUNCTIONAL HIERARCHY

The higher levels of the control hierarchy involve the planning and scheduling of interactive functions which are implemented in detail at the lower levels of the structure. These planning and scheduling tasks are well suited to the application of expert system techniques. Reference 4-4 provides an example of work on spacecraft command sequence generation that is directly applicable to level 0-1 executive control tasks (Figure 3-1). A wide range of sequencing and scheduling problems will exist that are very complex, are not necessarily determ- inistic, are subject to detailed operations constraints, and may require frequent rescheduling as mission requirements change or equipment faults force changes to preplanned sequences. These tasks are characterized by the need to coordinate the control of interactive lower level resources and by the need to perform the activity at a point in time prior to the actual detailed execution.

The lower levels of the hierarchy provide detailed control over specific functions in real or near-real time. Sensors can be provided to directly measure the operating status of the task and provide a local control resource with a more unambiguous determination of the health of hardware/ software in operation than is possible at the higher levels. In many cases, faults can be detected at these low levels and corrected without significant impact on system operation or the functioning of other subsystems. If a fault does have impacts outside the bounds of the lower-level control resource, the resource can identify the fault to the other affected resources to prevent its propagation as a fault in their operation. There will exist a set of faults that will require very high level co-ordination of resources for solution. This specialized set of faults has system level impact and is an appropriate area of concern for resources at levels 0-1 of the control hierarchy. Identification of this set of system level faults serves to limit the knowledge and resource demands upon expert systems at those functional levels.

The time domain requirements of the functional hierarchy are a significant design point. Some functions at levels 0-1 are significantly non-real time in nature. Examples are planning and scheduling system control sequences for the next day or hour's operation and analyzing the performance of a subsystem over a past period to predict future behavior. Real to near-

real time tasks involve the control, caution, and warning interface with the crew, system response to faults with system impact, and the fault tolerance of the executive control resources themselves. These types of tasks have significantly different requirements for timeliness of control response. The appropriate location of expert systems techniques in the system must address both the functional nature of the tasks and the performance requirements in the time domain.

Reference 4-3 describes a partitioning of ES and conventional control resources within a teleoperation system. The higher level control tasks progress from a "strategic" level which plans the execution of lower level tasks to the direct control of specific functional subtasks by distributed "tactical" control resources. This distribution of functions by time domain and control resource in a functional hierarchy is characteristic of current ES application for processes in real-time control.

4.7 SELECTING FUNCTIONS FOR EXPERT SYSTEM APPLICATIONS

Prior topics have characterized ES applications as being most valuable in planning and scheduling functions at high levels of the control architecture. A more complete characterization might be for problem solving at levels 0-1 or for extremely complex functions carried out at lower levels. Proximity operations, controlled by the guidance, navigation, and control subsystem, might contain such a potential application. It is possible to propose ES applications to control of many functions. However, the utility of such applications depends upon a trade between the complexity and resource requirements of an ES and the potential for a more economical traditional implementation. Control applications that do not involve a wide range of possible planning/scheduling factors, or that are not frequently utilized, are generally not candidates. Likewise, applications that can be easily automated with conventional software techniques, may not be worth the effort required to provide an ES application on-board.

SECTION 5

HARDWARE ASPECTS OF AUTONOMOUS CONTROL

A glance at the candidate functional control architecture of Figure 3-1 reveals a wide range of hardware involved in its implementation. This section discusses a series of topics related to hardware requirements analysis and selection.

5.1 DIVERSITY OF COMPUTING RESOURCE REQUIREMENTS

Functional divisions of the control architecture reveal highly diverse data processing and management requirements. Some specific requirements that vary widely with the functional application are:

(1) Floating point computation.

(2) Computation/logic throughput.

(3) Data intercommunication bandwidth.

(4) Symbolic processing application.

(5) Addressable volatile memory.

(6) Non-volatile memory availiability and usage.

(7) Self-checking, fault-tolerant operation.

(8) Real-time control requirements.

(9) Control display interfaces.

(10) External data bus communication bandwidth.

The variety of requirements at different levels of control make it difficult to satisfy all with one specific variety of processor or computer.

5.2 EXECUTIVE CONTROL REQUIREMENTS

The most significant issue of top level executive control is the recognition of this class of functions and the provision of hardware resources to support them. Several important design issues are dependent upon the detailed requirements for these functions. Some specific issues are: 1) selection of computer type for the executive functions; 2) provision of a single resource for the functions versus distribution among several; and 3) the automation of functions to best utilize crew resources. The following issues relate to the generation of more detailed requirements.

Executive control functions (level 0) are critical to coordination of the interactive functions of the system. Manned systems, such as the space station, have critical crew/ground interfaces at this level (control, display,

caution and warning, and system monitoring). The distribution of control resources in the hierarchical architecture can reduce the need for high - bandwidth data communications requirements amoung executive functions and lower level functions. There will still remain a need for significant processing capability for the functions that support the crew/ground control interface, top-level planning and scheduling, and automated system-level response for faults with major system impacts.

The planning and scheduling functions are major candidates for application of AI/ES technology. This implies that there may be requirements for symbolic processing computer resources and larger than normal volatile and non-volatile memory resources (subsection 4.3). These functions may well be the primary driver for sizing memory resource requirements for level 0 functions. Most other functions that could require significant amounts of mass storage (engineering data archiving, station data base maintenance, etc.) are more likely to be provided at level 1 or 2 in the Data Management Subsystem (DMS).

Ready access to engineering status telemetry for crew monitoring could entail a significant data communication rate requirement unless it is archived in the DMS and merely requested for special format displays by the executive functions.

5.3 AUTONOMOUS HEALTH AND MAINTENANCE IMPLICATIONS

Requirements to evolve new operational capability in flight and to automate fault detection and redundancy management have significant implications for the design of status measurments. These status measurements have traditionally been intended for inclusion in a telemetry stream of limited content and data rate. The measurements are frequently commutated in a manner that conceals time order relationships amoung different measurements. This characteristic of the telemetry process and the non-real time nature of the analysis process increase the difficulty of identifying which values are symptoms of a primary fault and which reflect the interactive response of components which have not failed. Providing carefully designed autonomous health and maintenance capability can significantly reduce the complexity of support for fault detection and analysis as well as aiding the process of testing new additions to the system.

The design process for hardware status measurements (telemetry points) should consider the real time nature of automated monitoring and fault response. Key measurements that directly reflect a failure within an element should be designed into the element as well as those performance measurements that permit trend analysis. Measurements should be unambiguous with respect to internal faults and allow the determination of conditions which may be caused by external faults.

Telemetry status measurements from the lowest levels of the control hierarchy should be software-reprogrammable at some level of control or data management resource. This allows hardware elements to be added, measurements to be changed as a function of operating mode, and displays to be augmented in abnormal circumstances. Basic hardware for the telemetry must be provided in the design of the low-level elements, but a higher-level selection process should allow for changes in desired data collection strategies.

Test data points and status monitoring points that are designed into elements for ground acceptance testing in manufacturing should be accessible in flight. This capability may rarely be required for any single element, but the programmability of the telemetry stream contents would allow the capability to be exercised when needed without constraining the resources for normal operations. Major elements such as earth sensors frequently have these test points designed into the hardware already, but they are simply not connected for flight. The ability to select or deselect these points at a local resource level can make them availiable if required.

In-flight testing may require that redundant elements be activated and exercised with real or simulated inputs while their alternates perform normal support. This will require special design provisions to inhibit logical or command outputs from the element under test from reaching operational elements. The design must also enable injecting non-operational test inputs and analysis of the response of the element under test.

5.4 FAULT TOLERANCE FOR CONTROL RESOURCES

Computing and data communications resources are subject to both transient and hard faults. Solid state memory and processors are subject to random faults caused by cosmic ray hits, electrostatic discharges, and noise sources. Incorporation of new technology such as Very Large Scale Integration (VLSI) may increase vulnerability to these effects. Hardware in long-duration missions, such as space station (20 years + on orbit), will be constantly exposed to this environment and the increased use of processors and memory in complex systems will increase the number of exposed components. Consistent, reliable operation of automated control systems will require fault tolerant design to reduce adverse impacts of these environmental effects.

Self-checking, fault-tolerant computers can offer continued operation in the presence of most hard and transient faults. A fault-tolerant machine will utilize time resources in restoring operation after occurrence of a fault, but offers continued operation and notifies the overall system (and crew) of the occurrence of the fault and the action taken. This is comparable to the best possible result in a non-fault tolerant machine which would result in some anomalous behavior of the system. The anomaly would require external intervention to restore operation, would consume significant personnel resources to investigate, and might not be traceable to the point of origin. A worst case would propagate to other subsystems. There are a number of design approaches to fault-tolerant computers. The most promising prospects utilize a combination of hardware sensing and reaction with software support (Reference 5-1). Hardware support provides rapid reaction to critical internal faults and reduces the use of memory address space as overhead for fault tolerance.

Hamming code techniques supported by redundant memory planes offer protection against memory faults with some overhead in data-transfer times. Transient "bit flips" can be corrected in the majority of incidental cases. Hard faults that fail to respond to bit resets can be replaced by spare resources. Audit trail requirements result in the identification of the fault action and a record of the fix and the resulting hardware configuration.

Data communications on internal intercommunications buses and external data buses are also prone to occasional faults. The relatively low speed data transfer requirements of distributed control allow a separate control data bus with lower error rates. Such a bus is also easier to protect from faults than the high-speed buses required for customer payloads and such specialized space station requirements as color video.

Fault-tolerant hardware design is becoming a standard application for high-value commercial applications such as telephone switching and aircraft avionics. Relatively little use has occurred in space applications though technology development programs for fault-tolerant computers have been under way for a number of years. Programs such as the space station will require significant increases in computing capacity over and above that required by past programs. Front end costs of providing fault-tolerant computing can be more than repaid in terms of operational reliability and reduced risk of failure in the control of complex systems (References 5-2 and 5-3).

SECTION 6

SOFTWARE ASPECTS OF AUTONOMOUS CONTROL

Autonomous control relies on a combination of hardware and software for implementation. The flexibility of control response availiable with software leads to a significant increase in the amount of on-board software in complex spacecraft systems. This increases the need to apply modern software design and engineering practices to increase productivity of the development process. Otherwise there will be a tendency to replace a ground-based "marching army" of operations controllers with a similar force of programmers.

6.1 SOFTWARE VERSUS HARDWARE RESOURCES

Hardware implementation best supports mature, well-understood tasks which do not change significantly over the life of the system. Sensing the state of components, implementing component state changes, and providing redundant functional capability are generic examples of such tasks. Software is best used to control new and complex tasks which have a significant risk of change or which will require flexibility for future growth. Software provides an inherent "robustness" of design that allows for adaptability to unexpected failures or design problems. The use of "read only memory" (ROM) or "firmware" for task implementation should be considered in the same light as hardware implementation due to the added difficulties of changing or overriding logic stored in ROM.

6.2 RESOURCE MARGINS IN SOFTWARE DESIGN

Memory size, processing throughput, and data communications capability are critical resources to be managed in the development process. Memory has a tendancy to be used if it is availiable, regardless of how large the amount may be. The Galileo Project established a software management policy that stated specific memory margins would be maintained at major design review milestones. This policy did not prevent the design from oversubscribing resources. This has led to a descoping of the software requirements to allow implementation in the fixed resource available. A complex project will be able to use how ever much memory is available; consequently, high-density memory and processors with large or virtual address spaces will be required to support the resulting designs. Processing throughput and data bus traffic rates must also be controlled through margin management policies. They are more amenable to design tradeoffs and distribution of functions to other control levels than the memory resource, however.

6.3 AN "OPERATING SYSTEM" FOR THE SPACE STATION

The autonomous control architecture example for the space station presented in Reference 2-1 provides a hint at the complexity of the hardware/ software system required for a large project. The referenced work addressed the functional control on the machine side of the man/machine interface. The complexity of the result raises the critical issue of how the crew and ground operations personel will control and interface with the station machine. The interface will involve display and control hardware driven by software. The

software will fulfill a function much like that of a traditional "operating system" for a computer. It must provide a control interface "language" for the operator to express his instructions to the system. The most visible requirement is for the operating system to allow operator monitoring and control over manual and automated tasks at all levels of the control hierarchy. This, however, is only part of the operating system function. Software routines will support operator display and communication, high-level performance analysis tasks, caution and warning, and a series of information utility tasks that will become apparent only as the operability of the station design is addressed.

6.4 COMMON FUNCTIONAL REQUIREMENtS FOR TEST AND OPERATIONS

The same fundamental operator interface and control functions needed for flight operations are required for integration and test of the system. Testing requires more lower-level data in more detail and with more frequency than nominal flight operations, but the functional tasks needed to request, process, analyze, and display the data are essentially identical. Requirements for integration and test of new or upgraded system elements in flight will cause the more detailed data and control requirements to arise at intervals in the mission. The test mode capabilities will also prove invaluable in support of failure analysis and anomaly investigations.

A properly designed operating system may be used throughout the life of the spacecraft system, from manufacturing test to flight operations. As the primary means of controlling the spacecraft, system operability requires that the operating system be a major system design consideration. This means that the requirements and design activity for this operating system proceed in parallel with (or precede) the design of the spacecraft. As a minimum, the major functional elements of the operating system need to be defined and a development plan must be derived that relates the schedule to the system design, implementation, and test schedule.

6.5 A COMMON "TEST AND OPERATIONS CONTROL LANGUAGE"

The "language" which the operator uses to express control actions and operating procedures to the machine part of the system is critical to the ease and reliability of use of the system. Computer operating systems (Unix, CP/M, MS/DOS, VM/OS, etc.) each has its own language of commands and associated syntax which offer varing degrees of "user friendliness" to the operator. The real power of the operating system is in the software functions that are executed as a result of interpreting the inputs expressed in the language. The design of the language is critical to the operator`s ability to understand and use these software functions in a reliable manner.

An important distinction to note is that this "language" is an interface and control language for operator communication with the machine. It is not a programming language such as ADA, HAL, or FORTRAN. Indeed, these programming languages would be candidates for writing the software that implements the Test and Operations Control Language (TOCL). The language will, however, allow the development of prestored batch procedures to accomplish complex predetermined sequences of functions. In this respect, it will provide a high-level "system programming" capability.

This language concept itself is not new in the aerospace community. The "System Test and Operations Language" (STOL) was developed at the University of Colorado to support the Solar Mesosphere Explorer flight project. It has been utilized for spacecraft system test, itegration of spacecraft with ground operations, and flight operations. A "European Test and Operations Language" (ETOL) is utilized by the European Space Research Organization (ESRO) as its standard for control interface with its spacecraft. The Kennedy Space Center has developed a system called GOAL for the automated prelaunch checkout of the space shuttle. There is currently an effort to establish an IEEE standard for such languages that draws upon the experience of all these efforts and the general automated testing technology community.

The crucial point for the space station project is that the same standard should be utilized for all phases of the project. Several possible benefits could result:

(1) The language represents a unifing concept in the design of control for the station at all levels, among diverse contractors, and of the different NASA centers.

(2) Test procedures at subsystem level and below can be passed on to integration test and maintained to support flight tests for performance validation, integration of new capabilities, and fault investigation.

(3) Resources would not be expended to develop and implement procedures in contractor-unique systems that are not portable to system operations support.

(4) Personnel participating in integration and test are allowed to gain operations experience that can be applied in flight operations for normal operations or anomaly investigations and contingency operations.

Specification and design of this language is a significant part of the operating system implementation. It should be a front end intensive activity to support development and test of early versions before the major system design activity takes place, as initial versions must be availiable at least to support subsystem level test activity.

6.6 SOFTWARE DESIGN FOR OPERATIONAL FLEXIBILITY

Software must offer maximum flexibility with minimum impact upon configuration control and retesting. The data-base-driven logic designs should be used as much as possible to allow changes to specific operating behavior without impact upon compiled code. This concept is discussed in subsection 4.4 as a move towards expert system implementation. Regardless of this issue, it provides the most reasonable means of modification of software behavior and operating features by operations support personnel and crew.

6.7 AUDIT TRAIL REQUIREMENTS

Reference 1-2 discusses the concept of an audit trail. This is a record of the control actions carried out by an autonomous or automated system. The information identifies the time history of control logic execution, circumstances that initiated the actions, and the resulting states of the system. An audit trail supports system test, validation of operational behavior, and monitoring of performance by operations personnel. The concept was originally proposed to allow the action of automated fault protection software to be traced after the fact of execution. It also allows visibility into the actions of any automated function. Programmability of audit trail contents allows the basic storage resource to be utilized at different levels of detail and for a variety of different purposes at different times in the mission. The audit trail process may be compared to the debug feature offered in the compilers of many programming languages.

SECTION 7

ISSUES IN AUTONOMY IMPLEMENTATION

Autonomous control influences the design, test, and operations of space systems. Our lack of experience in system level autonomy raises a number of issues of concern. This section introduces and comments on some of these issues.

7.1 DESIGN AND IMPLEMENTATION METHODOLOGY

Implementation experience is limited to small numbers of critical functions selected on the basis of special mission requirements. Valuable lessons have been developed from this experience, but there is no experience with an integrated, system-level design approach for autonomous control. Preferred architectural approaches have been developed at a functional level (Reference 2-1) through studies and analysis activities and tentative approaches to a design methodology have been proposed (Reference 1-2). These are preliminary efforts, however, and the current approaches to system level design for autonomous control remain a matter of engineering judgement extrapolated from previous flight experience.

7.2 CONTROL ARCHITECTURE

Designing a system with a complex set of autonomous functions leads to difficulties in timing and resource contention with a central computer implementation. There are many potential decentralized resource control architectures that may be selected for a particular mission. A hierarchical functional description of the system is useful in the design process, but there are many constraints in the next important step of extending the functional design to a specific allocation of functions to resources in subsystems. Individual designers and organizations will have preferred approaches based on extensions of previous work. There remains an overall lack of hard experience with the trade-offs involved in the initial architecture allocation to hardware resources.

7.3 COMPONENT AVAILABILITY

Availability of flight-qualified components is a major barrier to design of an autonomous control architecture. The design concepts that have been developed utilize large numbers of computing resources, significant quantities of random-access memory, fault-tolerant hardware, high-capacity mass memory, and a significant degree of control software. Components which meet functional characteristics such as fault tolerance, or which have acceptable mass and power usage for their performance, are not readily availiable. This constrains the designers at both system and subsystem levels to designs which significantly limit the number of autonomous functions or compromise the system level implementation of the design.

7.4 SYSTEM DESIGN EXPERIENCE

The three issues above tend to skew cost/risk trades of partitioning functions towards ground implementations or force initial autonomy requirements to be descoped to fit within availiable resources. Deliberate choices for potential onboard functions and architectural options must be tested in prototype form to reduce perceived risks and develop a proven implementation methodology. Major components with operating characteristics that support a low risk implementation (fault-tolerant, low mass/power consumption, etc.) must be developed and proven. Finally, a demonstration of the utility and functioning of system level autonomous control features must be performed to prove the basic concept and develop implementation experience.

7.5 TEST AND VALIDATION

This may prove one of the most difficult problems of the entire field. Frequently it is impossible to test all states of a conventional complex system. The problem will be more difficult with significant degrees of automated or autonomous control. Designing for minimum functional interaction, complete testing at lower levels of the design, and automation of the test process with a common test control procedural language offer some approaches to ease test and validation problems. Many significant challenges remain. How does one choose a set of functional tests to validate the integrated system performance even if complete testing has been accomplished at lower levels of the design? How does one design a fault-tolerant system with the capability to inject simulated faults to exercise design features without impacting nominal operation? How does one decide what set of regression tests must be run on the system when it has been updated with new components?

Use of AI or ES techniques may further complicate the validation problems. J. Matijevic notes that a proof of a mathematical theorem related to four-coloring planar maps was developed by an ES at the University of Illinois in the late 1970's. The proof has not yet been accepted by much of the mathematics community, however, as the proof method is not "observable". The same difficulty may arise with the validation and test of some AI techniques in sequence generation and control. Acceptable techniques must produce results in a verifiable manner.

SECTION 8

REFERENCES

1-1. Turner, P. R., "Autonomy and Automation for Space Station Housekeeping and Maintainance Functions", ASME Winter Meeting, 17 November 1983.

1-2. Turner, P. R., "Autonomous Spacecraft Design and Validation Methodology Handbook", Issue 1, USAF Space Division Report SD-TR-82-58, JPL D-188, 30 April 1982.

1-3. Space Station Task Force, "Space Station System Operational Requirements", Baseline Issue, December 1983.

2-1. "Autonomous Spacecraft Systems: Architecture and Technology", JPL D-1197, 15 December 1983.

4-1. Nau, D. S., "Expert Computer Systems", IEEE Computer, February 1983.

4-2. Engelman, C., et.al, "KNOBS: An Integrated AI Interactive Planning Architecture", AIAA Computers in Aerospace IV Conference, Hartford Connecticut, 24-26 October, 1983.

4-3. Orlando, N. E., "A System for Intelligent Teleoperation Research", AIAA Computers in Aerospace IV Conference, Hartford Conn., 24-26 October, 1983.

4-4. Vere, S. A., "Planning Spacecraft Activities with a Domain Independant Planner", AIAA Computers in Aerospace IV Conference, Hartford Conn., 24-26 October, 1983.

4-5. Friedman, L., "Transforming Expert-System Software Into Hardware", Jet Propulsion Laboratory, Pasadena, CA 91109, January 1984

5-1. Riethle, G. S., "Self-Checking Computer Module", JPL D-1121 for Autonomous Redundancy and Maintenance Management Subsystem Demonstration Project, September, 1983.

5-2. Rennels, D. A., "Fault-tolerant Building-Block Computer Study", JPL Publication 78-67, July 1978.

5-3. Avizienis, A. A, Ercegovac, M. D., and Rennels, D. A., "Fault-tolerant Computer Study, Final Report", JPL Publication 80-73, February, 1981.

APPENDIX A

TOPICAL SUMMARY

The purpose of this appendix is to condense and highlight significant points from each topical discussion in the text. The key points within the discussion and rationale are extracted and presented as they might be for a viewgraph presentation.

1. AUTONOMY DEFINITION AND ATTRIBUTES

   a. Autonomy is a system attribute created by design of system functions and the means for autonomous control.

   b. Characteristics of Autonomy are:

      (1) Control is maintained within the defined boundary of the system.

      (2) Fault-tolerant operation within the system is required.

      (3) A specified level of autonomous performance must be maintained for a specified time duration.

      (4) Adaptability to changes in internal or external conditions are met by sensing and analyzing the conditions, directing a change to the system operation, and implementing the required response.

      (5) Human participation in the process is allowable if man is included in the boundry of the defined system.

   c. Autonomous system control is fundamentally a hierarchical process.

   d. Machine autonomy is a combination of automation and fault tolerance.

   e. Functions requiring autonomous control include those which are interactive with other functions within the system and those which may be controlled independently of other functions.

   f. Time aspects of autonomy include both the duration of the required autonomous operation and the real-time nature of the control.

   g. Computer and data system technologies that support fault-tolerance and provide more performance for lower power, mass, and risk, are fundamental to practical autonomous control implementations.

2. AUTONOMY AND COST/BENEFIT ISSUES

   a. Addition of autonomy has a front-end cost impact upon a program that is more visible and more easily assessed than the potential long term benifits.

b. Cost and complexity issues related to autonomy must be contrasted with cost, complexity, and ability to perform the mission without autonomy.

c. Many control design features of a complex system must be provided regardless of the autonomy of the design. Such items should not be charged against autonomy.

d. Top-down system and subsystem level design of autonomous functions in a complex system is more effective in cost and performance than later "patching" individual subsystem level design features.

e. Life-cycle payoffs for autonomy require the deliberate planning of operations for reduced ground support and project-level commitment to meet the goal.

f. Initial sizing of resource requirements for autonomous control should include a significant margin as the complexity of implementation of requirements is often greatly underestimated.

g. Traditional reliability models are not applicable to analysis of systems containing autonomous redundancy management and maintenance.

3. SYSTEM ARCHITECTURE FOR AUTONOMOUS CONTROL

a. Control of a complex system composed of multiple subsystems is fundamentally a hierarchical and distributed control process.

b. Implementation of a distributed control hierarchy involves both data processing (logical control) and data communications (of sensory data and commands).

c. Distribution of computing resources in the control hierarchy can reduce the contention of parallel functions for real-time resources.

d. Fault-tolerant computing in the control hierarchy can protect the system against the propagation of failures in the controllers and can monitor the system to simplify analysis of faults.

e. Self-checking fault-tolerant computing may be applied at higher executive levels of the control hierarchy to protect a wide range of lower-level functions or devices, reducing the need for self-checking computers at those levels.

4. AUTONOMY AND ARTIFICIAL/MACHINE INTELLIGENCE

a. Expert systems (ES) technology provides the most promising AI field for early applications in system control.

b. The selection of appropriate domains for expert systems is critical to successful applications.

c. Potential implementation problems caused by inability to provide ES hardware and software resources in a flight environment can be addressed by restricting the application domain.

d. Modern software design techniques of data-base driven logical control and modular programming techniques can be naturally extended toward ES implementations.

e. Expert systems are most appropriately placed at executive levels of the control hierarchy.

5. HARDWARE ASPECTS OF AUTONOMOUS CONTROL

a. Diversity of computing requirements in the control hierarchy suggests a need for different types of processors in the design.

b. Executive control levels should be afforded the protection inherent in fault-tolerant self-checking computers.

c. Requirements for real-time autonomous control have significant impacts upon hardware and software architecture and design.

d. Several hardware and software supported fault-tolerant design features are availiable to protect system operation fron transient and permanent faults in the control resources.

6. SOFTWARE ASPECTS OF AUTONOMOUS CONTROL

a. Hardware control implementation is well suited for mature, well understood tasks which do not change significantly over the life of the system.

b. Software is well suited for new and complex tasks which have a significant risk of change or which require flexibility for future growth.

c. Memory size, processing throughput, and data communications loading are critical resources affecting control software.

d. Critical resource margins must be established and managed to prevent over-running capacities with attendant impact upon software implementation.

e. The space station machine requires a software-based "operating system" to implement crew and ground control.

f. Operating system requirements for integration test and flight operations are fundamentally the same.

g. A "Test and Operations Control Language" (TOCL) is required as the interface between man and the operating system.

h. The TOCL is not a programming language, though procedures may be written in it in a manner similiar to high level programs.

i. The TOCL can serve as a powerful tool to enforce common understanding of control design standards in a diverse community of contractors participating in the program.

j. An "audit trail" of autonomous control actions is required to provide visibility into the operation of the machine and to support test and validation.

7. ISSUES IN AUTONOMY IMPLEMENTATION

a. Prior flight experience has led to current design concepts, but they have not been validated by demonstration.

b. The hierarchical functional architecture has many conceptual advantages for complex systems, but allocation of the functions and interactions between computing resources needs design and testing experience.

c. Lack of flight-qualified components to implement an architecture limits the development of concepts by experienced system designers.

d. Test and validation requirements for complex autonomous systems may prove a major barrier to implementation.